

REMARKS

Applicant appreciates the courtesy and consideration of the examiner in a recent telephone interview with the undersigned attorney for applicant. However, applicant believes that the examiner misunderstands the nature and scope of applicant's invention and the cited prior art. After all, applicant's invention is a software development system for dynamic server side web applications, while the prior art is a method for editing static web sites. Applicant submits that software development systems and editors for static documents are two quite different application domains.

The examiner apparently believes that the method disclosed in the D'Arlach patent can be used to develop web applications. However, while the editor of D'Arlach is indeed a web application, the editor is only able to edit static web sites, not applications.

The issues as presently framed for appeal are:

- A. Whether claims 26, 27, 30-33, 43, 67, 69, and 71-72 are patentable under Section 102(e) over U.S. Patent No. 6,151,609 to Truong.
- B. Whether claims 1-8, 22-24, 28, 29, 41, 42, 51-66, 68, 70, 73-96, and 114-127 are patentable under Section 103(a) over the combination of Truong and U.S. Patent No. 6,026,433 to D'Arlach.
- C. Whether claim 25 is patentable under Section 103(a) over the combination of Truong, D'Arlach and U.S. Patent No. 6,651,108 to Popp et al.

Applicant submits that all pending claims are patentable over the cited references for the reasons detailed below.

I. OVERVIEW OF THE INVENTION

Applicant has developed a unique editor for server-based ***dynamic web applications***. A server-based ***dynamic web application*** is a software program that generates web pages upon browser requests. In general, a user visiting a web site running a ***dynamic web application*** receives different versions of the same web page when viewing it multiple times. For example, a web page with a shopping basket changes depending on the number and kind of items that the user has put into the basket. At first, the user will see a version of the page with an empty shopping basket; later on, the user will receive versions of the page with items placed inside the shopping basket.

In contrast, a *static web site* is a set of documents that are delivered unchanged to the user's browser and which remain unchanged until changed by the author.

Applicant's invention uses "page templates" and "components" to build server based *dynamic web applications*. Instead of programming a web application from scratch, the idea is to create the application by connecting preexisting software components. Applicant's page templates are used to specify how the components are to be connected to each other and to the layout. While the user visits the web application, page templates are automatically transformed into the generated pages and sent to the browser for display to the user.

For example, the server computer might contain a page template with a shopping basket component. When the user requests the page template, the shopping basket component inserts the actual content of the shopping basket into the page template leading to a newly generated page for display to the user.

WYSIWYG editing of dynamic web applications is a complicated problem, because pages dynamically change during execution of the application. Thus, there is no single page view that could be shown to a developer for editing. Applicant's invention addresses this problem by showing a running application in the editor. This problem is not addressed in the art cited by the examiner: The Truong patent edits source text only and does not follow the WYSIWYG principle and the D'Arlach editor is for static web pages only.

Advantageously, in applicant's editor the application looks similar to and functions like the normal running application as viewed by the end user. Conventional editing technology does not really address this issue – in general, page templates look different than the generated page as viewed by the user. In contrast, applicant's editor runs the application during editing and shows the generated pages to the developer while actually applying changes to the page template. Using applicant's editor would cause the display of the example page to have a functional shopping basket thereby allowing the developer to add and/or remove items from the basket and to see the end user's actual view during editing.

II. SUMMARY OF ARGUMENTS

In rejecting the claims, the examiner relies primarily on the editor disclosed in the Truong patent, either by itself or in combination with features of the D'Arlach patent. However, Truong's editor shows only the source code of the application during editing, which looks very

different to the end users' view, and the source code is not functional during editing. D'Arlach is not directed to an editor *per se*, but instead to a method for using templates to create **static web sites**. Notwithstanding the examiner's assertion, there is no teaching or suggestion in D'Arlach that D'Arlach's method could be used to edit server-side **dynamic web applications**. Further, neither Truong nor D'Arlach teaches running a functional applications during editing.

The examiner suggests that the templates disclosed in D'Arlach are the same as claimed here but a key difference is that applicant's templates include at least one component that generates browser code while a user visits the web application. In addition, applicant's editor contains specific functions to add, modify, and delete software components, and it works on components that generate browser code and/or operate on the server side. Truong's editor does not have these functions – it only describes browser built-in components. Such components do not operate on the server and do not generate browser code. D'Arlach's elements on the other hand are data items stored in a data base and not executable software components. Thus, neither Truong nor D'Arlach teach or suggest the use of components having instructions to generate browser code, nor do they teach or suggest components running on the server.

III. ARGUMENTS

A. Claims 26, 27, 30-33, 43, 67, 69 And 71-72 Are Patentable Over Truong

The examiner asserts that claims 26, 27, 30-33, 43, 67, 69 and 71-72 are anticipated by Truong. However, applicant respectfully traverses the rejection. The Truong patent was discussed in detail in applicant's last two responses, which are incorporated herein by reference.¹

In sum, Truong discloses a web-based text editor which does not permit real time editing of fully functional, running web applications, as the examiner has now acknowledged. (See Paper No. 18 at pp. 4-5).

1. Independent Claim 26 is Patentable

The examiner rejected independent claim 26 on the basis that Truong at column 3 lines 27-38 discloses a first software program including instructions for transforming a first document into a second document having features which permit editing of the first document such that at

¹ See Amendment and Remarks in Response to Office Action dated February 28, 2003, and Amendment and Remarks in Response to Office Action dated October 27, 2003

least part of the second document appears and functions similar to the first document. In applicant's reading of the cited text, Truong explains the display of an editor selection form which lists filenames for the user to select for editing. The examiner explicitly mentions the editor selection form, apparently as the second document. Since the editor selection form lists the filenames only, but not the file content, it can not possibly appear or function similar to the document being edited. In contrast, applicant's claim explicitly requires that the second document appear and function similar to the first document.

Other parts of Truong, for example, column 3 lines 40-42 and column 10 lines 40-50, discuss functions for editing source text. Source text, however, does not appear similar to the normal view of a document and it does not function at all because it is not executed. This is also shown by Truong's figure 5. If the application is not functional at all during development, then it certainly does not function similar to the first document. In contrast, applicant's claim requires that "at least a part of the second document appears and functions similar to the first document." Truong simply does not do that.

In addition, with reference to D'Arlach, the documents specified in claim 26 are clearly different from D'Arlach's site templates. D'Arlach's site templates are stored as objects and elements inside a database and are therefore not integral documents.

2. Dependent Claims 27, 30-33, and 43
Are Patentable Over Truong

Claims 27-33 and 43 are dependent from claim 26, and are patentable over the cited reference for all the same reasons.

With regard to dependent claim 27, the examiner cites Truong at column 8 lines 13-15 as disclosing a component being executed by the first software program. The cited portion refers to the editor input form, which is a browser built-in HTML form. In contrast, applicant's claim requires that the component be executed by the first software program that does the document translation. Truong makes no such teaching or suggestion.

With regard to dependent claim 30, the examiner cites Truong at column 7 lines 1-5. However, the cited portion appears to describe simply that a browser can execute scripts. In contrast, applicant's claim explicitly requires that scripts may be used as *"features which permit editing of the first document such that at least part of the second document appears and*

functions similar to the first document” Truong fails to teach or suggest that scripts could be used in this way.

With regard to dependent claim 31, the examiner cites Truong at column 9 lines 15-20. In applicant’s reading, the cited portion discusses using scripts to do local processing. In contrast, applicant’s claim specifically requires that the scripts transport information encapsulated in the first document to the browser. There is no such teaching or suggestion in Truong.

With regard to dependent claim 32, the examiner cites Truong at column 9 lines 10-15. However, the cited portion refers to an editor input form which shows only the source text. In contrast, applicant’s claim requires “*features which permit editing of the first document such that at least part of the second document appears and functions similar to the first document.*” The source text of a web page certainly looks drastically different from its normal view. Thus, Truong also does not teach or suggest such features.

With regard to dependent claim 33, the examiner cites Truong at column 7 lines 10-30. In applicant’s reading, the cited portion explains the general operation mechanism of a web browser. It does not specifically refer to change requests. In contrast, claim 33 explicitly requires the information incorporated into the second document be used “*to send change requests for the first document to the server.*” Truong does not teach or suggest this feature.

With regard to dependent claim 43, the examiner cites Truong at column 3 lines 30-40 and column 9 line 18. The first cited portion appears to refer to various actions without specifying if they are done by a script or by the normal browser function controlled by HTML. The second cited portion appears to refer to scripts contained in the second document. In contrast, applicant’s claim 43 requires “*at least one additional script that works in cooperation with the second document.*” There is no such teaching or suggestion in Truong.

3. Dependent Claims 67, 69, 71 and 72
Are Patentable Over Truong

Applicant notes that claims 67, 69, 71, and 72 are dependent from claim 59, which was rejected based on the combination of Truong and D’Arlach. Thus, it is inconsistent and improper to reject these claims on Truong alone. However, for all the same reasons that claim 59 is considered patentable, these claims should also be considered patentable.

With regard to claim 67, the examiner cited Truong at column 8 lines 39-50 as disclosing that the view, except for editing features, looks similar to the end-user view of the generated document. However, the cited portion discloses the file selection form but does not give information on similarity. On the other hand, Figure 5 clearly indicates that Truong's editor shows the source code of any document. The source code of an HTML document does not look similar to its end-user view.

With regard to claim 69, the examiner cited Truong at column 3 lines 35-38 as disclosing that the editor uses the edit information to modify the dynamic web document. However, the cited portion appears to refer to an editor selection form used to select a file for editing. The cited portion does not appear to relate to modifying a dynamic web document or using edit information, and D'Arlach does not supply the missing features.

With regard to claim 72, the examiner cited Truong column 7 lines 59 to 67 as disclosing initiating a reload in the browser. In applicant's reading, the cited portion discloses loading of the remote editor program, however, it does not seem to disclose initiating a reload in the browser. A reload in the browser means that the browser will load the same page a **second time**. In contrast, the cited portion discloses a **first time** load.

B. Claims 1-8, 22-24, 28, 29, 41, 42, 51-66, 68, 70, 73-96, and 114-127 Are Patentable The Combination Of Truong And D'Arlach

The examiner asserts that claims 1-8, 22-24, 28, 29, 41, 42, 51-66, 68, 70, 73-96, and 114-127 are obvious over the combination of Truong and D'Arlach. However, applicant respectfully traverses the rejection.

1. D'Arlach
 - a) Overview

D'Arlach introduces a "Method of creating and editing a web site in a client-server environment using customizable web site templates." D'Arlach's method appears to be applicable for editing static web sites, but not for editing server side web applications. D'Arlach uses web site templates to simplify web site development for the author. Instead of developing a complete web site anew, the author is given a template as a starting point. Then the user customizes the template and finally publishes the result. Internally, D'Arlach's templates consist

of objects and elements, and as taught by D'Arlach, these objects and elements are actually data items stored in a database and not software components.

b) Editing Web Sites

The use of the words "Web Site" in the title of D'Arlach already suggests that D'Arlach's editor is intended for Web Sites and not for server side dynamic web applications. While D'Arlach's editor itself is a server side dynamic web application, this by no means suggests that his editor can edit such applications. In fact, in column 4 lines 14–16, the D'Arlach patent describes how a dynamic web application works, and in the same sentence makes clear that his invention is useful to maintain web sites, which suggests that D'Arlach sees a clear distinction between a web site and a software program.

According to column 5 lines 30–33, during publishing, D'Arlach's editor generates a set of web pages that make up the web site. Since users visit the web site only after publishing, this means that generation actually takes place before the visit and not during the visit. Server side dynamic web applications, on the other hand, generate pages while the user visits the web site. Thus, D'Arlach's editor produces web sites, but not server side web applications. Further, applicant's claims make clear that pages are generated while the user visits the web site, i.e., upon request by the browser. For example, claim 1 recites "*A software development system for applications that run on a data network ... whereupon request by the browser, an application generates generated documents for display by the browser and responds to the request with the generated documents pages.*" Claim 59 recites "*A software development system for developing dynamic web documents, said dynamic web documents operating by being transformed into an end user's view upon a request by a web browser, the end user's view being provided to the browser in response to the request.*" Claim 74 recites "*A software development system for document templates that are intended for transformation into generated documents ... a document generator comprising second instructions to, upon document requests, generate generated documents from at least one document template for display by the first software program wherein the set of components on the generated documents can vary for different document requests for said document template.*"

c) Site Templates

D'Arlach introduces web site templates to enable users to easily and efficiently maintain a web site. In order to create a web site, the user selects a pre-built web site style template and then customizes objects or elements contained therein, as described in column 4 lines 54-67. When customizing is finished, the web site is published and static web pages are generated. In contrast, applicant uses the term "templates" to refer to dynamic page templates that are evaluated per page request and used to create dynamic web applications.

The basic idea of D'Arlach is to present a web site author with a pre-built template for customization. In contrast, using applicant's software development system, authors build templates from scratch and the templates are processed automatically while users visit the web application. In fact, in the cited portions of D'Arlach there is no discussion of edit operations for adding arbitrary elements to a site, while operations of customizing the preexisting template are explained in detail.

D'Arlach's templates are intended to give the author an advantage to more easily and more efficiently create web sites. Applicant's templates allow the author to create dynamic and interactive web applications instead of static web sites.

D'Arlach's templates are customized by the author of a web site before publishing, while Applicant's templates are processed automatically while a user visits the web application.

d) Components

D'Arlach's elements are not "components" as that term is used by applicant, but instead merely data items stored in a database. There are only two types of elements taught in D'Arlach as described in column 5 lines 1-15 and support for them is built directly into the editor and the database schema. They do not contain instructions to generate browser code and are not executed on the server while a user visits a web site. In D'Arlach's invention, generation takes place at publishing time as described in column 5 lines 30-33.

In contrast, applicants components contain instructions for execution on the server while a user visits the web site. Users can create web applications using applicant's editor by placing components on page templates and thereby connecting them. This distinction is explicitly reflected in applicant's claim language. For example, claim 6 recites "*server computer further comprising . . . a plurality of components ... including at least one component that reacts*

interactively on user input by executing instructions contained in said component . . .” Claim 22 recites *“a document generator for ... executing components.”* Claim 51 recites *“a plurality of components containing instructions to generate browser code.”* Claim 74 recites *“a plurality of components that include instructions to generate browser code.”* Claim 114 recites *“a plurality of components for execution on the server.”* Claim 125 recites *“running the application thereby executing selected components and generating a generated document.”*

e) Editing Functional Applications

Since D’Arlach’s editor works on web sites, it has no need for any features that keep an application functional during editing. The cited prior art does not appear to reveal such features. In contrast, applicant’s invention includes a page generator that executes the application and, when used during editing, collects edit information and generate edits features while executing the application, handles etc.

D’Arlach’s editor, on the other hand, does not even show web sites in a functional way: clicking on an element representing a link during editing does not seem to operate the link as required for a functional web site, but instead edits the link-element. This is discussed in detail with regard to claim 96. In contrast, applicant’s editor shows handles the user can click on for editing and thereby keeps links in working order.

f) Database Use

D’Arlach internally uses a database to store the web site templates. All edit operations are done performing database updates, so D’Arlach’s editor can be seen as a database application. Web pages are then generated from the database. Whenever the user wants to change the web site, changes are actually done on the database and pages generated again.

In contrast, applicant’s page templates are stored as integral documents in files just as ordinary web pages or program source code. This advantageously allows users to also use standard programs, e.g., source code editors, on these files. This distinction is expressed in claim 26, which recites, *“a system to modify documents on a server in a data network which couples said server computer to a client computer.”* In addition, D’Arlach’s editor is internally structured as a database application, which is much different than a document processing application.

g) Other distinctions

D'Arlach's editor operates by generating pages for display by the browser. As shown in figure 2, it operates on the server only. In contrast, applicant's editor operates partly on the server and partly inside the browser on the client. Advantageously, this allows many editing operations to be done without communicating with the server and consequently editing operations are much faster. This distinction is expressed in claim 90, which recites "*the editor including a first software program for execution within the browser and for processing selected clicks on the view of said document displayed in the browser by initiating editing functions.*"

h) The Examiner's Reasoning

The examiner acknowledges that "Truong doesn't explicitly disclose an editor capable of directly operating on the documents displayed by the browser thereby allowing the user to work on a functional application during development." (Office Action dated August 13, 2004 at pp. 5). However, the examiner alleges that D'Arlach discloses analogous art "creating or editing a working copy of a user's site with the option of publishing the updated modified page or creating a new web site." However, applicant submits there is no analogy because D'Arlach does not teach or suggest how a user would edit a functional application. As explained above, D'Arlach works on static web sites only so there is no functional application to edit.

The examiner further writes on page 5 that D'Arlach would "enable a working copy or user web page to be modified dynamically." Applicant assumes that this refers to D'Arlach's option of publishing a modified page. However, publishing always requires the web designer to manually start the publishing operation. See, for example, D'Arlach at column 5 lines 20-25. Thus, D'Arlach's modification has nothing to do with a web application that generates new pages for each client request. In contrast, applicant's invention contains specific features to edit web applications.

In his reasoning towards claim 2, the examiner further acknowledges that "Truong doesn't explicitly disclose wherein[sic] developed applications comprising document templates or editing components on templates and executing components on page templates." (Office Action at p. 5) However, the examiner asserts that D'Arlach discloses analogous art. D'Arlach's objects and elements are, however, just data items (*see* column 5 lines 14-16) and can not be executed. In contrast, applicant's components are software components intended for execution.

Applicant submits that his amended claim language adequately recites key differences allowing the claims to distinguish over the cited art.

Advantageously, applicant's software development system is made for editing dynamic server side web applications. In contrast, D'Arlach's editor just works on static web sites as explained above.

i) Combining With Truong

The examiner fails to make clear how D'Arlach would combine with Truong. First, it is unclear how the combination would provide an end user's view during editing. D'Arlach's templates require a database, which Truong processes text files only. Neither reference runs the application during editing. Further, neither reference teaches or suggests components that run on the server, nor components having instructions that generate browser code. Thus, applicant submits that the combination is improper, and in any event, it fails to teach or suggest these key features of the present invention.

2. Independent Claim 1 Is Patentable Over Truong Combined With D'Arlach

Claim 1 stands rejected based on the combination of Truong and D'Arlach. However, applicant respectfully traverses the rejection.

The examiner cited Truong column 10 lines 45 to 50 as disclosing a page generator running an application being developed and sending generated documents to the browser. However, Truong's page generator runs only the editor, not the application being developed. The cited portion explicitly states that the text of the file being edited is sent directly to the browser, without actually executing the file. Also, figure 5 of Truong shows that the file being edited is not run during editing.

The examiner acknowledges that Truong does not explicitly disclose an editor capable of directly operating on the documents displayed by the browser thereby allowing the user to work on a functional application during development. However, the examiner asserts that D'Arlach discloses analogous art by creating or editing a working copy of a user's web site with the option of publishing the updated modified page or creating a new user web site. The cited portion does not teach or suggest that pages during editing look or function similar to the final pages, nor that

pages are functional at all during editing. In contrast, applicant's claim requires a functional application during development.

D'Arlach does not edit a dynamic web application, but just a static web site. This is indicated by the language used by D'Arlach, i.e., "web site," and also by the fact that page generation in D'Arlach takes place upon publishing a web site (column 5 lines 30-32), while web applications require generation of pages per request. Because D'Arlach operates on web sites and not on applications, it is not analogous art since it doesn't teach or suggest any way to keep a web application functional during editing.

Since neither Truong nor D'Arlach provides a functional application during editing, and claim 1 requires a functional application during development, applicant submits that claim 1 is patentable over the combination of D'Arlach and Truong.

In addition, applicant submits that it is not possible to combine D'Arlach and Truong because Truong edits text files and D'Arlach's templates are stored in a database. Thus, it is not possible to apply Truong's editor on D'Arlach's templates or vice versa.

For all the foregoing reasons, applicant submits that claim 1 as pending is patentable over the cited combination.

Applicant has amended Claim 1, not to define over the cited art, but to further clarify the term application: "*A software development system for applications that run on a data network ..., the applications upon request by the browser generating generated pages for display by the browser and responding to the request with the generated pages*". As explained above, D'Arlach describes a method of creating web sites. In contrast, claim 1 requires a software development system for applications that upon request by the browser generate pages.

3. Dependent Claims 2-5, 41, and 42 Are Patentable Over Truong Combined With D'Arlach

Claims 2-5, 41, and 42 are dependent from claim 1, and are patentable over the cited combination for all the same reasons.

In addition, with regard to claim 2, the examiner cites Truong at column 2 lines 1-5 as disclosing components, giving text boxes and buttons as examples. Column 2 mentions HTML elements that are part of the browser. In contrast, claim 2 requires components to be executed by the page generator.

The examiner further cites column 10 lines 47 to 50 as disclosing features to insert, modify and delete a component. In applicants reading the cited portion however just introduces insert, modify and delete operations for characters, not for components.

The examiner acknowledges that “Truong does not explicitly disclose developed applications comprising document templates or editing components on templates and executing components on page templates.” The examiner writes that D’Arlach discloses analogous art and cites D’Arlach at column 5 lines 25 to 45. As previously described, D’Arlach’s templates are generated at publishing time (see column 5 lines 30 to 33), while claim 1 requires templates to be generated upon each browser request. Applicant’s templates work very differently and have a different purpose when compared to the site templates of D’Arlach, as explained above. In addition, the objects and elements disclosed in D’Arlach are data items (see column 5 lines 14 to 16) stored in a database and can not be executed. In contrast, claim 2 requires components to be executed.

The examiner states that it would have been obvious to combine Truong and D’Arlach because using templates would enable users to create and maintain web sites easily and efficiently, and cites D’Arlach at column 4 line 60 to 63. In contrast to D’Arlach’s templates, applicant’s templates are primarily used to create dynamic web applications and not merely to simplify development of static web sites.

Claims 3-5 are dependent on Claim 2, and for all the same reasons, applicant submits that claim 3-5 are not taught or suggested by Truong, either alone or in combination with D’Arlach.

With regard to claim 3, the examiner cited Truong column 2 lines 1-5 as disclosing that at least one of the components reacts interactively on user input by executing instructions on the server. However, the cited portion actually describes HTML elements that are capable of sending information to the server. In contrast, claim 3 requires that the component themselves react and execute instructions on the server. In Truong, the HTML elements are interpreted by the browser and are not actually present on the server, which also means that they cannot execute instructions on the server.

With regard to claim 4, the examiner cites D’Arlach at column 4 lines 64 to 65. The cited portion describes that elements have attributes or properties associated with them. In contrast, the claim requires one component to contain at least one other component. According to claim 2,

components are items to be executed by the page generator. In contrast, the elements, attributes, and properties disclosed in D'Arlach are not executable items.

With regards to claim 5, the examiner cites figure 5 item 506 of D'Arlach. Step 506 refers to selecting a template for creating or editing a site. The cited portion does not teach or suggest generating different pages for different requests. As explained above, D'Arlach generates pages during publishing, not upon a user request. This means that a page, once published, stays the same for each page request. In contrast, claim 5 requires a varying set of components for different requests of the same page template.

With respect to claims 41 and 42, the examiner cites Truong at column 3 lines 30 to 38 and specifically refers to the editor input form and the editor selection form. However, the editor selection form is used by Truong to select a file for editing. The cited portion does not teach or suggest that the editor selection form is an executable program. In contrast, claim 41 requires a client part for execution on the client computer. Likewise, the cited portion does not show any executable instructions. In contrast, claim 42 explicitly requires instructions. Applicant's editor has a client part consisting of many instructions that is downloaded to the client computer when the editor is started. Applicant has amended claim 42, not to make a patentable distinction, but to clarify that the client part is indeed used for the editing itself.

4. Independent Claim 6 Is Patentable Over Truong Combined With D'Arlach

Claim 6 stands rejected based on the combination of Truong and D'Arlach. However, applicant respectfully traverses the rejection.

Claim 6 is an independent claim directed to a software development system that allows the user to create client server based applications by plugging together components. Applicant's invention provides components that can interactively communicate with the user on the client computer and that can execute instructions on the server as well. Claim 6 as pending explicitly requires components that operate on the server, reciting "at least one component that reacts interactively on user input by executing instructions *contained in said component on the server.*" Applicant had previously amended the claim by adding the limitation "contained in said component." The examiner's comments on page 8 of the previous office action do not appear to address this amendment; neither do the comments on page 6 of the current office action. Instead, the examiner cited Truong at column 2, lines 1 to 5, as disclosing a plurality of components

residing in the data store on the server including components that react interactively on user input by executing instructions on the server. However, Truong refers to browser built-in HTML elements, like HTML forms and HTML fields. Since these components are built into the browser, they reside in the data store of the client and they are executed on the client, not the server. HTML forms and fields can send data to the server, but they do not contain instructions for execution on the server. This should be clear since the complete browser with everything in it runs on the client only. In contrast, the claim clearly requires components to contain instructions on the server.

The examiner also cites Truong at column 5, lines 60 to 65. This cited portion explains how internet servers send URLs and pages to each other and then finally to a client. The cited portion does not, however, teach or suggest components as recited by applicant

The examiner also cites Truong's Figure 3c item 128 as disclosing a data store. The cited step includes the legend "*store each string as a variable*," which indicates only the fact that data is stored, and not the fact that the data store contains components. This has been acknowledged by the examiner.

The examiner further acknowledges that Truong does not explicitly disclose a plurality of document templates residing in the data store, at least one document template and a third program selecting a document template based on the request from the client computer and instructions for generating a document for the document template for transmission to the client computer. The examiner asserts that D'Arlach discloses analogous art, namely a template database. However, applicant submits that D'Arlach generates pages from his template database at publishing time, not per request, as explained in D'Arlach at column 5 lines 30-32. Thus, D'Arlach does not have a third program for selecting a document template based on the request from the client computer.

In addition, D'Arlach does not have software components on page templates, since the objects and elements are merely data items stored in a database. Consequently, these objects and elements do not contain instructions for execution on the server as required by the claim. Also, the objects and elements of D'Arlach can not react interactively on user input by executing instructions on the server because D'Arlach generates pages at publishing time, i.e., before the user actually interacts with the generated pages.

Applicant therefore submits that because neither Truong nor D'Arlach teach or suggest components containing instructions for execution on the server claim 6 is patentable over the combination of Truong and D'Arlach.

5. Dependent Claims 7 and 8 Are Patentable Over Truong Combined With D'Arlach

Claims 7 and 8 are dependent from Claim 6, and for all the same reasons, applicant submits that claims 7-8 are likewise not taught or suggested by the combination of Truong and D'Arlach.

With regard to claim 7, the examiner refers to his reasoning in rejecting claim 5. However, claim 5 deals with a varying set of components. In contrast, claim 7 requires “*instructions for interactively editing selected components.*” As pointed out with respect to claim 2, Truong’s editor does not have a specific function for “*interactively editing selected components*” - it just has a function for editing characters of the source code. As explained above, D'Arlach also fails to teach or suggest software components.

With regard to claim 8, the examiner cited D'Arlach at column 4 line 60 to 65 as teaching components inside templates and objects/properties or attributes within a component. In applicant’s reading, the cited portion refers to site templates, elements, objects, properties and attributes, but not to components. According to the requirements given in claim 6, components must contain instructions for execution on the server. Neither templates, elements, objects, properties nor attributes contain instructions for execution on the server - all of these items are merely data items stored in a data base as explained in D'Arlach at column 5 lines 15 to 16. In addition, according to claim 6, components are incorporated inside page templates, which means that page templates do not qualify as components. Attributes and properties are not individual objects and therefore not components. Since the templates, elements, objects, properties and attributes are not components in the sense of the claim, D'Arlach does not teach or suggest nested components as claimed.

6. Independent Claim 22 Is Patentable Over Truong Combined With D'Arlach

Claim 22 stands rejected based on the combination of Truong and D'Arlach. The examiner, however, did not cite any text portion from D'Arlach. In previous actions, the examiner rejected the claim based on a combination of Truong and Fleskes. Applicant respectfully traverses the rejection.

The examiner cited Truong at column 11 line 17-20 as disclosing an editor operable with the web browser for inserting, deleting, and modifying components on document templates. However, the cited portion discloses that the WINDOWS editing commands, such as copy, insert, delete, and paste, may be used if the browser is running on the WINDOWS operating system. The cited portion does not teach or suggest that the editing commands work on components and it does not teach or suggest a modify operation as required by the claim. In contrast, at column 10 line 47, Truong writes "the text may be edited at the browser using editing features such as delete, select, search, copy, paste, and the like." This makes it clear that Truong's editing features work only on text, and not on components.

The examiner also cited Truong at column 10 line 45 to 50 as disclosing a document generator for processing document templates, executing components and for generating documents from the document templates that are understandable by the web browser. In applicant's reading, the cited portion discloses providing the text of the selected file to the web browser. The cited portion does not teach or suggest components or a document generator for executing components, as required by the claim.

Claim 22 requires a document generator executing components. As explained above D'Arlach's elements are data items and therefore not executable. Browser built-in HTML tags as mentioned by Truong are not executed by the document generator. Applicant therefore submits that claim 22 is patentable over Truong and D'Arlach.

7. Dependent Claims 23-24 Are Patentable Over Truong Combined With D'Arlach

Claims 23-24 are dependent from claim 22, and for all the same reasons, applicant submits that Claim 23 is likewise not taught or suggested by Truong, either alone or in combination with D'Arlach.

With regard to claim 23, the examiner cited Truong at column 2 lines 1 to 5 and column 10 lines 45-50 as disclosing that the editor operates functional applications in an edit mode thereby permitting editing directly in the web browser. However, the cited portion actually discloses editing of source text in the web browser, but not the operation of functional applications in an edit mode. In fact, Truong at column 10 lines 45-50 states that the editor edits the source text and Figure 5 shows an example. In contrast, applicant's editor operates a functional application in an edit mode thereby permitting editing of generated pages directly in the browser.

With regard to claim 24, the examiner cited Truong at column 10 line 45 to 50 as disclosing a component that can react on subsequent document requests by executing selected instructions. The cited portion actually describes instructions of the editor. In contrast, the claim requires that the instructions belong to the component and be executed upon subsequent document requests.

8. Dependent Claims 28-29 Are Patentable Over Truong Combined With D'Arlach

Claims 28-29 depend from claim 26, and for all the same reasons, applicant submits that claim 28-29 are likewise not taught or suggested by the combination of Truong and D'Arlach.

Further with regard to claim 28, the examiner cites D'Arlach at column 3 lines 33 to 45 and refers to selecting and editing templates which include editable components and objects. In applicant's reading, the cited portion is an introductory remark not referring to page templates or selecting. In particular, the cited portion does not specify how an element is selected nor does it mention anything like a handle. In applicant's reading of D'Arlach, there are no handles, but instead, elements are selected by directly clicking on them. As described in column 9 lines 13 and 14 of D'Arlach, clicking on the "Phone Services" button itself, and not on a handle, brings up the "Elements Properties" screen. In addition, figure 10 does not show anything like a handle. In contrast, applicant's claim explicitly requires handles.

9. Independent Claim 51 Is Patentable Over Truong Combined With D'Arlach

Claim 51 stands rejected based on the combination of Truong and D'Arlach. based on the same reasoning as applied to claim 6. However, applicant respectfully traverses the rejection.

Claim 51 is an independent claim describing a system for editing components on web document templates. The main distinction between Truong's editor and the claimed invention is that Truong does not have components and that his editor shows source text only. This distinction is represented in the claim language, for example "*a user interface for editing functions used for maintaining components on document templates.*" The cited references do not teach or suggest a user interface having editing functions for maintaining components. The editing functions actually disclosed in Truong are clearly identified as working on text by stating: "the text may be edited at the web browser using editing features." (See Truong at column 10, lines 47 to 50) In addition, Figure 5 of Truong shows the user interface of the editor, but does not show "*a user interface for editing functions used for maintaining components on document templates,*" as recited in claim 51.

In his reasoning with regard to claim 6, the examiner cited Truong as disclosing a plurality of components. However, as discussed above, Truong discloses browser built-in HTML forms and HTML fields. Since these items are built into the browser, they do not encapsulate or generate browser code as required by the claim.

Applicant amended claim 51 to recite "*a plurality of components containing instructions to generate browser code.*" In contrast, D'Arlach's elements do not contain instructions but are data items stored in a data base.

Applicant therefore submits that because neither Truong nor D'Arlach teach or suggest components containing instruction to generate browser code claim 51 is patentable over the combination of Truong and D'Arlach.

10. Dependent Claims 52-58 Are Patentable Over Truong Combined With D'Arlach

Claims 52-58 are dependent from Claim 51, and for all the same reasons, applicant submits that claims 52-58 are likewise not taught or suggested by Truong, either alone or in combination with D'Arlach.

With regard to claim 52, the examiner cited Truong column 7 line 1 to 15 as disclosing components including fourth program instructions including steps to generate browser code prior to transmission to the browser program. The cited portion describes web browsers that can execute scripts. Scripts can indeed be used to generate browser code, but inside the browser. The examiner cited Truong at column 2 line 1 to 5 as disclosing components in the discussion of claim 2 and claim 6, identifying HTML textboxes and buttons as components. However, HTML text boxes and buttons do not contain scripts for generation of browser code. In addition, HTML element and scripts are clearly part of the browser. In contrast, the claim explicitly requires the generation of browser code prior to the transmission to the first software program. D'Arlach's elements are data items and therefore do not contain instruction including steps to generate browser code. Applicant amended claim 52 to clarify the fact that different requests can result in different browser code being generated.

Claim 55 is directed to edit functions working on components. The examiner refers to his reasoning towards claim 6. However, the cited portion of D'Arlach does not appear to teach or suggest the operations of adding a component or removing a component from a document template as required by the claim.

With respect to claim 56, the examiner cites D'Arlach's figure 5 number 506. According to column 5 lines 38 to 40, step 506 deals with selecting a site template, not with elements or components. Claim 56, however, requires the calling of fourth program instructions. Fourth program instructions are introduced in claim 52 as instructions included inside at least some components to generate browser code prior to transmission to the first software program. As explained above, D'Arlach's elements are merely data items not containing any instructions. In contrast, claim 56 requires instructions included in some components to be called.

With respect to claim 58, the examiner cited D'Arlach column 10 lines 15 to 25 as disclosing clicking on the generated document. The cited portion of D'Arlach, however, seems to disclose clicking various buttons in order to manage complete web sites. The term "the generated document" used in claim 58 refers back to claim 56, saying that the generated document is generated from a document template. The menu buttons cited by the examiner are part of the editor and are not generated from the template.

11. Independent Claim 59 Is Patentable Over Truong Combined With D'Arlach

Claim 59 stands rejected based on the combination of Truong and D'Arlach based on the same reasoning as applied to claim 6. However, applicant respectfully traverses the rejection.

Claim 59 is an independent claim directed to a software development system for dynamic web documents capable of displaying functional applications during editing. Applicant submits that claim 59 is quite different than claim 6 because it is specifically concerned with editing technology, and therefore the reasoning for claim 6 is wholly inapplicable to claim 59.

As explained above, Truong's editor works on the source code of an application and consequently does not follow the WYSIWYG principle and does not execute the application during editing. In contrast, the inventive editor is capable of executing the application being developed during editing and so the application appears functional during editing. This distinction is expressed by the following claim language, "*generated documents ... which look and function similar to the end user's view of the documents*" and by "*the editor program comprising first instructions for requesting the document generator to process a dynamic web document leading to a generated document.*" This language makes clear that the editor indeed executes the application during editing. Additional claim language, namely "*instructions to modify the dynamic web document*" make clear that the dynamic web document that is being edited is the one being requested.

D'Arlach's editor can not edit dynamic web documents, but is used for static documents only. As explained above, document generation in D'Arlach happens during publishing, whereas dynamic web documents are generated when users visit the application. Applicant amended claim 59 to set forth a clear definition of the term dynamic web document "*A software development system for developing dynamic web documents, dynamic web documents operating by being transformed into their end user's view upon a request by a web browser and the end user's view being provided to the browser in response to the request*".

12. Dependent Claims 60-66, 68, 70, and 73 Are Patentable Over Truong Combined With D'Arlach

Claims 60-73 are dependent from Claim 59, and for all the same reasons, applicant submits that Claims 60-73 are likewise not taught or suggested by Truong, either alone or in combination with D'Arlach.

With regard to claim 61, the examiner cited Truong column 8 lines 39 to 50 as disclosing further instructions for execution during document generation to collect edit information for use by the editor. The cited portion describes document generation as part of the editor itself. According to claim 59, "the document generator" means the document generation usually required for display of a dynamic web document. In contrast, the cited portion of Truong is concerned with document generation taking place inside the editor.

With regard to Claim 63, the examiner states that Truong does not explicitly disclose automatically requesting that the document generator process the dynamic web document if required. However, the examiner states that D'Arlach discloses analogous art, citing item 620. In applicants reading of D'Arlach, column 6 lines 8 to 20 describes a CGI program that transmits the updated page. Since D'Arlach's method does not repeat the original request, URL parameters of the original request for the page get lost. Therefore, D'Arlach's method does not work in general for dynamic web documents. Claim 59, however, requires the editor to work for dynamic web documents.

With regard to claim 64, the examiner cites Truong column 9 lines 15 to 20 as disclosing components including instructions for use by the document generator to generate browser code. The cited portion refers to javascript code for execution inside the web browser. It does not teach or suggest anything about components. In contrast, applicant's claim explicitly requires instructions for use by the document generator and instructions contained in components.

With regard to claim 66, the examiner refers to his reasoning towards claim 2. In addition to the reasoning of claim 2, the cited portions of D'Arlach do not appear to reveal the operations of adding a component to a document template as required by the claim.

With regard to claim 68, the examiner referred to Claim 61 for reasoning, and applicant likewise refers to his response to that reasoning towards claim 61, as discussed above.

With regard to claim 70, the examiner referred to Claim 64 for reasoning. Claim 70 requires "*position information on the components contained in the document template*" while

claim 64 does not. Applicant therefore submits that Truong does not teach or disclose position information and therefore is not relevant for claim 70.

With regard to claim 73 the examiner cites D'Arlach column 10 lines 15 to 25. In applicants reading the cited portion deals with managing complete web sites and not with elements of web pages. According to D'Arlach columns 6 lines 4 to 6 D'Arlachs editor requires a server interaction in order to display information on an element. In contrast, claim 73 requires information on an element to be displayed without requesting the document generator.

Applicant has amended claim 73 in order to correctly state that requesting the document generator refers to any document, not the generated document.

13. Independent Claim 74 Is Patentable Over Truong Combined With D'Arlach

Claim 74 is an independent claim describing a software development system using components on web document templates. The examiner rejected this claim using his reasoning towards claim 6. Applicant also refers to the discussion of claim 6 above. Truong's editor works on text only, while applicant's editor contains specific editing functions for handling components. This distinction is represented in the claim language, namely "*an editor capable of performing edit functions maintaining components on document templates.*" Truong's editor is not capable of performing edit functions maintaining components. The editing functions disclosed in Truong column 10 lines 47 to 50 clearly identify themselves as working on text "*the text may be edited at the web browser using editing features*".

In addition, claim 74 explicitly requires components to generate browser code using the claim language "*a plurality of components, that include instructions to generate browser code for transmission to the first software program.*" In his reasoning with respect to claim 6, the examiner cited Truong column 2 lines 1 to 5 as disclosing a plurality of components. However, Truong refers to browser built-in HTML forms and HTML fields. Browser built-in components do not generate browser code for transmission to the browser program as required by claim 74.

The claim explicitly requires document templates to be dynamic, stating that the set of components on the generated documents can vary for different document requests for the same document template. D'Arlach creates web sites, in contrast to web applications, and generates

pages while publishing, not per request, as explained above. This means that according to D'Arlach, a page can change only by modifying the template and publishing it again.

Furthermore Claim 74 requires "*components that include instructions*". In contrast, D'Arlach's elements are data items and therefore do not contain instructions.

Applicant submits that because neither Truong nor D'Arlach teach or suggest components containing instruction to generate browser code claim 74 is patentable over the combination of Truong and D'Arlach.

Applicant amended claim 74 to fix the inconsistent use of the singular and plural tense and not to define over the cited art.

14. Dependent Claims 75-89 Are Patentable Over Truong Combined With D'Arlach

Claims 75-89 are dependent from claim 74, and for all the same reasons, applicant submits that claims 75-89 are likewise not taught or suggested by Truong.

With regard to claim 75, the examiner cited D'Arlach column 10 lines 10 to 25 and lines 50 to 60. The cited portions refer to managing web sites and web pages, and not to components as required by the claim. In addition, the cited portions of D'Arlach do not reveal the operations of adding a component to a document template as required by the claim.

With regard to claim 76, the examiner cited Truong column 6 lines 57 to 63 as disclosing that tag syntax is used to denote components on document templates. The cited portion reveals only that HTML tags are denoted using tag syntax on document templates. However, HTML tags are not "components" as defined by applicant since, according to claim 74, components generate browser code prior to transmission to the first software program, and also, components cooperate with the editor program. This is not the case for HTML tags.

With regard to claim 78, the examiner cites D'Arlach column 10 lines 10 to 35 as disclosing excluding a component that can react interactively on subsequent document requests from the generated document. As discussed with regard to claim 74, D'Arlach's elements can not be interpreted as components because they are merely data elements, not software components, and do not contain instructions. In particular, the component that can react interactively as required by claim 78 can not be interpreted as an element, because elements are data items that can not react. Further, the cited portion talks about editing the web site template and thereby

removing an element. The claim, however, refers to a web application that, for selected requests, excludes components, e.g., a shopping application that hides a shopping basket in case it is empty. The claim language expresses this by the use of “the generated document” referring back to claim 74. Applicant has amended claim 78 not to better define over the cited combination, but to further clarify this claim language.

With regard to claim 79, the examiner cites Truong figure 3b item 164 and related text. The cited portion refers to handling of the ABORT button of the editor. The claim, however, deals with the components on the document templates created with the document development system. Applicant therefore submits that the cited portion is irrelevant as applied to this claim.

With regard to claim 80, the examiner cited D’Arlach column 5 lines 23 to 25 and “storing of the customized template limitation”. Claim 80 is dependent on claim 78, and as reasoned with regard to claim 78, these claims refer to excluding components while an application is running without changing the template. In addition, there is no teaching of session memory or of any information stored in session memory. Applicant amended claim 80 to correct the use of “*the generated document*,” not to provide a patentable distinction over the cited combination.

With regard to claim 81, the examiner cited D’Arlach column 4 lines 60 to 65. The cited portion deals with manually editing a web site template. The claim, however, refers to a web application that, for selected requests, excludes certain components, e.g., a shopping application that hides a shopping basket in case it is empty. The claim language expresses this by the use of “the generated document” referring back to claim 74. Applicant has amended claim 81, not to further define over the cited combination, but to further clarify this claim language. In addition, the cited portion does not teach or suggest anything like a first component containing sixth instructions to decide about exclusion. In fact, D’Arlach’s elements are data items and do not contain instructions.

With regard to claim 82, the examiner cited Truong column 3 lines 30 to 35 as disclosing an editor taking the varying set of components into account. In applicant’s reading, the cited portion describes the editor input and editor selections form used to select a file for editing and does not seem to provide any information on how the editor shows a particular file. However, Truong’s Figure 5 clearly indicates that the editor shows the source code of an application being edited. In contrast, the claim requires the editor be able to take the varying set of components

into account, which implies that the editor must be capable of showing various views of the same document.

With regard to claim 83, the examiner cited D'Arlach column 4 lines 60 to 65. The cited portion deals with manually excluding elements from a page template. The claim, however, refers to editing a web application that, for selected requests, excludes certain components. In applicant's editor, an application appears functional during editing and can include or exclude components while functioning. In contrast, D'Arlach's editor is for static web sites, and pages change only when they are manually changed by the author. The claim language expresses this by requiring an editable view that includes and excludes components similar to the end user's view of the document. In contrast, D'Arlach's editor just shows a static web page that changes only if the user asks the editor to modify the template. Applicant has amended claim 83, not to further define over the cited combination, but to further clarify this.

With regard to claim 84, the examiner referred to claim 4 for his reasoning. However, claim 4 is concerned with nested components. In contrast, claim 84 is concerned with the set of components on the document template and on the generated document. Since claim 5 is concerned with a the set of components, applicant submits that for all the reasons given with claim 5, Claim 84 is likewise not taught or suggested by Truong. Applicant amended claim 80, not to provide a further distinction over the cited combination, but to correct the use of "*the generated document*".

With regard to claim 85, the examiner cites D'Arlach column 6 lines 35 to 45. The cited portion deals with changing options from the set of available options inside the editor. In contrast, the claim requires the generated document to contain multiple instances of a component contained in the template. For example, in applicant's invention, a shopping basket component has multiple instances and could contain multiple products, depending on the number of articles actually placed into the basket. The use of the term "the generated document" refers back to claim 74. Applicant amended claim 85, not to provide a further distinction over the cited combination, but to further clarify this.

With regard to claim 86, the examiner cited Truong column 8 lines 20 to 35 as disclosing instructions to qualify names generated into the browser code with the unique identifier. The cited portion seems to disclose including file names into a generated page. However, the file names are known before the generation process. In contrast, claim 86 requires that a unique

identifier be assigned to each component instance, and that this unique identifier be used to qualify the names.

With regard to claim 87, the examiner cites D'Arlach column 4 line 40 to 35. In applicant's reading, the cited portion describes the basic operation of D'Arlach's editor as a CGI program processing page requests. The cited portion does not seem to disclose elements, components, instances of components, nested components or templates. Instead, the cited portion deals only with the operation of the editor, but not with the operation of the site templates. In contrast, the claim requires that the software development system work on templates that, for selected requests, display multiple instances of a component in the template on the generated page. The claim language expresses this by the use of "the generated documents" referring back to claim 74. Applicant has amended claim 87, not to provide further distinction over the cited combination, but to further clarify this claim language.

With regard to claim 88, the examiner refers to his reasoning of claim 85, and applicant traverses this rejection for the same reasons. In addition, applicant submits that D'Arlach does not have an editable view that displays multiple instances of a component, but instead, D'Arlach just shows a static web page that changes only if the user asks the editor to modify the template. Applicant amended claim 88 to remain consistent with amended claim 83.

With regard to claim 89, the examiner cites D'Arlach figure 6 item 604. In applicant's reading, the cited item refers to an element editing form. The cited portion does not show instructions contained in components for generating browser code, as required by the claim.

15. Independent Claim 90 Is Patentable Over Truong Combined With D'Arlach

Claim 90 is an independent claim. The examiner included a discussion of claim 90 twice, once together with the reasoning of claim 1, and a second time independently. These discussions seem to contradict each other since the first reasoning relies on D'Arlach while the second one does not cite D'Arlach at all.

In the second reasoning the examiner cited Truong column 1 lines 65-67, column 2 lines 1-10 and 17-30, as disclosing an editor that allows the user to edit a document being displayed by the browser. The cited portion does not reveal details about Truong's editor, but instead provides background regarding web technology, such as form capable browsers. Such browsers

can edit text contained inside form fields displayed on an HTML page. However, this is different from editing the HTML page itself. In fact, Truong column 10 lines 45-50 and Figure 5 reveal that Truong's editor is made for editing a file displayed inside a form field and not for editing the displayed HTML page itself.

The examiner also cited Truong column 7 lines 1-5 as disclosing scripts staying functional during editing. However, the cited portion reveals only that the browser is capable of processing scripts. This does not imply or teach or suggest that the file being edited is sent to the browser in such a way that the browser recognizes and executes the scripts during editing. In fact, Truong column 10 lines 45-50 discloses that the text of the file is sent to the browser in such a way that it can be edited as text, which implicitly means that scripts are not executed by the browser but are shown in source code form. In addition, Figure 5 of Truong shows editing of a script, which further supports applicant's contention that according to Truong, scripts in the selected file are shown as text and are not executed.

With respect to the examiner's adoption of his reasoning towards claim 1 relying on Truong and D'Arlach, applicant refers to his discussion of claim 1 above. In addition, applicant submits that the cited portions do not teach or show a first software program for execution within the browser and for processing selected clicks on the view of said document displayed in the browser by initiating editing functions. Figure 2 shows that D'Arlach's editor is a software program for running on the server only. Figure 2 does not show the editor including a first software program for execution within the browser on the client. In contrast, significant parts of applicant's editor are implemented as scripts for execution within the browser. Advantageously, this works much faster because less server interactions are needed. In column 6 lines 3 to 5, D'Arlach describes that the editor immediately requests the server when the user clicks. In contrast, the claim explicitly requires a first software program for processing selected clicks.

Truong uses browser built in text boxes for editing including associated program instructions. These can not be used and combined with D'Arlach, however, because scripts contained therein are displayed as source code and are not functional as required by the claim. Applicant therefore submits that because D'Arlach does not have a first software program as claimed and because the programs shown by Truong are not usable, claim 90 is patentable over the combination of Truong and D'Arlach.

16. Dependent Claims 91-96 Are Patentable Over Truong Combined With D'Arlach

Claims 91-96 are dependent from claim 90, and for all the same reasons, applicant submits that claims 91-96 are likewise not taught or suggested by the combination of Truong and D'Arlach.

With regard to claim 91, the examiner cited Truong column 6 lines 55 to 65 as disclosing the editor using a second browser window for showing information on elements contained in a document. However, the cited portion discloses HTML elements, but does not talk about browser windows. In addition, there are no details provided on the user interface of Truong's editor, merely information on the web browser used.

With regard to Claim 92, the examiner cited Truong column 10 lines 45 to 50 as disclosing modifying documents in cooperation with the first software program. However, the cited portion reveals transmitting the content of a file to the web browser. This is clearly different from modifying a document in cooperation with the first software program. Applicant has amended the claim to clarify that the modifications are also stored.

With regard to Claim 93, the examiner cited column 7 lines 20-30 as disclosing that a generated document looks similar to the original. The cited portion talks about generating a document for display to the user, but does not indicate that the generated document looks similar to any other document. On the contrary, Truong's Figure 5 shows clearly that the editor displays only the source code text, which looks, in the case of HTML documents, very different than the original view of an HTML document. Applicant has amended the claim to fix an inconsistent use of "the document" with respect to the base claim 90.

With regard to Claim 94, the examiner cited Truong column 2 lines 1-10 as disclosing instructions for generating browser code for components. The cited portion discusses various HTML elements, but does not disclose any generation process. Applicant has amended the claim to fix an inconsistent use of "the document" with respect to the base claims 90 and 93 and to clarify the definition of components.

With regard to claim 96, the examiner cites Truong column 8 lines 39 to 45 as disclosing that links contained in the document stay functional during editing. In applicant's reading, the cited portion refers to the file selection form, but not to the document being edited and not to

links. In applicant's reading, Truong's editor shows links during editing as source text, and not as functional links, as required by the claim. Also, D'Arlach's editor does not show functional links. As described in column 9 lines 13 and 14 of D'Arlach, clicking on the "Phone Services" button during editing does not operate the link as in the users view, but instead, brings up the "Elements Properties" screen. This makes browsing using the normal link navigation of a web site impossible during editing. In contrast, applicant's claim requires that links stay functional during editing in order to allow the user to browse and edit at the same time.

17. Independent Claim 114 Is Patentable Over Truong Combined With D'Arlach

Claim 114 is an independent claim. The examiner did not include specific reasoning for this claim, but instead referred to the same reasoning used to reject claim 1. However, this seems inappropriate because Claim 114 explicitly claims components that include features to cooperate with the editor. In contrast, Claim 1 does not recite anything to do with components at all.

In rejecting claim 6, the examiner discusses components. However, Truong discloses only browser built-in components, which can not generate browser code. In contrast, the claim requires "*components including second program instructions to generate browser code.*" In addition, browser built-in components run on the client as part of the browser. In contrast, the claim requires "*a plurality of components for execution on the server.*"

As previously discussed, D'Arlach only works for static pages. In contrast, the claim is about dynamically generated documents. In addition, D'Arlach's objects and elements are data items in a database and not software components. Thus, they do not contain any program instructions. In contrast, the claim requires at least one of the components to include instructions to generate browser code.

Applicant therefore submits that because neither Truong nor D'Arlach teach or suggest components for execution on the server claim 114 is patentable over the combination of Truong and D'Arlach.

18. Dependent Claims 115-124 Are Patentable Over Truong Combined With D'Arlach

Claims 115-124 are dependent from Claim 114. Thus, for all the reasons given with regard to claim 114, applicant submits that Claims 115-124 are likewise not taught or suggested by Truong either alone or in combination with D'Arlach.

With regard to claim 115, the examiner cited Truong column 10 lines 55-59 as disclosing that first features include instructions for passing information to the editor. According to claim 114, components for execution on the server include first features for cooperation with the editor. The cited portion reveals the use of instructions for sending a file back to the server. These instructions are a part of the editor for execution on the client computer. In contrast, claim 115 requires instructions to be part of the components for execution on the server computer.

With regard to claim 116, the examiner cited Truong column 7 lines 60-67 as disclosing that part of the information is collected during execution of the components on the server. However, the cited portion describes execution of the editor, not of the components. Truong does not teach or suggest any execution of components on the server. The examiner has apparently interpreted HTML text boxes and buttons (column 2 lines 1-5) as “components” in his discussion of claim 2. These HTML elements are part of the browser, however, and are not executed on the server. In contrast, claim 116 requires components to be executed on the server thereby collecting information for the editor.

With regard to claim 118, the examiner cited Truong's Figure 5 as disclosing that the information includes attributes of the component. However, Figure 5 of Truong shows the user interface for the editor. The figure does not provide any details about “the information” and therefore appears irrelevant.

With regard to claim 119, the examiner cited column 10 lines 45 to 50 as disclosing first features including fifth instructions that display additional editing features. The cited portions seems to reveal editing features as part of the editor and browser. In contrast, the claim requires first features to include fifth instructions to display the editing features, whereby the first features need to be included in the component according to claim 114, not inside the editor or the browser.

With regard to claim 120, the examiner cites item 602 and related text as disclosing handles. In applicant's reading, the cited portion talks about selecting an element but does not

specify how, nor does it mention anything like a handle. In applicant's reading of D'Arlach, there are no handles, but instead, elements are selected by directly clicking on them. As described in column 9 lines 13 and 14 of D'Arlach, clicking on the "Phone Services" button itself, and not a handle, brings up the "Elements Properties" screen. In addition, figure 10 does not show anything like a handle. In contrast, applicant's claim explicitly requires handles.

With regard to claim 121, the examiner cited column 10 lines 45 to 50 as disclosing first features including extensions for use by the editor. The cited portions reveal various features as part of the editor and browser. In contrast, claim 121 requires the first features to include the extensions, whereby the first features are included in the components according to claim 114, not inside the editor or the browser.

With regard to claim 122, the examiner cited Truong column 8 lines 65 to 67 as disclosing a web page for editing component attributes. The cited portion discusses the generation of web pages in general and does not disclose the generation of a web page for a specific purpose. In contrast, claim 122 specifically requires a page for editing component attributes.

With regard to claim 123, the examiner cited Truong column 6 lines 57-63 as disclosing components denoted on document templates using tag syntax. However, the cited portion discloses **HTML tags** denoted on document templates using tag syntax. **HTML tags** are different from **components** as used by applicant in claim 114, since components are required to run on the server and generate browser code. Applicant submits that the cited portion does not disclose components denoted on document templates using tag syntax.

With regard to claim 124, the examiner cited Truong column 8 lines 65 to 67 as disclosing components including instructions to generate browser code. The cited portion discloses instructions to generate browser code, but it does not specify where. Claim 124 requires that these instructions be part of the components.

19. Independent Claim 125 Is Patentable Over Truong Combined With D'Arlach

Claim 125 is an independent claim directed to a method for editing an application. The examiner did not include specific reasoning for this claim, but instead used the same reasoning as for claim 1. In the discussion of claim 1, applicant reasoned that neither Truong nor D'Arlach

taught or suggested running an application during editing. In addition, running an application during editing would make Truong dysfunctional. D'Arlach is not for developing applications at all, as explained above. In contrast, the claimed method for editing an application includes a step of running the application.

In addition, claim 125 contains limitations involving components. The examiner's use of the same reasoning applied to claim 1 seems misplaced since Claim 1 is not concerned with components at all.

In his reasoning toward claim 6, the examiner discusses components. However Truong discloses only browser built-in components, which are executed during document display. In contrast, claim 125 requires components to be executed during document generation prior to document display using two steps: a first step of "running the application, thereby executing components and generating a generated document," and a second step of "displaying a view of the generated document."

Claim 125 requires execution of components. In contrast, D'Arlach's elements are merely data items and not software components.

Applicant therefore submits that because neither Truong nor D'Arlach teach or suggest execution of components during document generation claim 125 is patentable over the combination of Truong and D'Arlach.

Claim 125 further requires a step of identifying a component in the source code of the application and modifying the source code of the application. D'Arlach does not appear to show such steps- in fact, D'Arlach's templates are stored in a database instead of using source code. Truong also does not teach or suggest an identifying step. Applicant therefore submits that claim 125 is patentable over the combination of Truong and D'Arlach.

20. Dependent Claims 126-127 Are Patentable Over Truong Combined With D'Arlach

Claims 126-127 are dependent from Claim 125, and for all the same reasons, applicant submits that Claims 126-127 are likewise not taught or suggested by Truong either alone or in combination with D'Arlach.

With regard to claim 126, the examiner cited Truong column 10 lines 45 to 50 as disclosing repeating the running and the displaying steps after applying a modification function.

However, applicant submits that the cited portion does not teach or suggest a step of running the application. With respect to claim 125, the examiner referred to steps 102 and 104 of Figure 3C as the running step. As clearly depicted in Figure 3C, these steps are not repeated. In contrast, claim 126 requires that the running step to be repeated.

With regard to claim 127, the examiner cites Truong column 8 lines 45 to 52 for collecting edit information for use by the identifying step. In applicant's reading of the cited portion, Truong teaches identifying files for editing. In contrast, the identifying step of claim 125 refers to identifying a component on a page template.

C. Claim 25 Is Patentable Under Section 103(A) Over The Combination Of Truong, D'Arlach And U.S. Patent No. 6,651,108 to Popp et al.

Claims 25 is dependent from Claim 24, and for all the same reasons, applicant submits that Claims 25 is likewise not taught or suggested by Truong either alone or in combination with D'Arlach and Popp.

Further, the examiner acknowledges that neither Truong nor D'Arlach discloses component classes implementing components, but bases his rejection on the Popp patent, stating that the use of classes is a well known general practice in an object oriented environment. However, Truong's browser built-ins are not executed by the document generator as required by claim 22. Further, D'Arlach's elements are data items stored in a database and not suited for execution as explained above. Thus, there would be no need to implement D'Arlach's elements by any technology. This means that a combination including Popp would be ineffective and improper.

The examiner further reasons that it would have been obvious to use Popp because the use of classes is a general practice in an object oriented environment. In applicant's reading, however, neither Truong nor D'Arlach use an object oriented environment. Thus, the examiner's reasoning is misplaced.

D. Claim 128 Is Patentable Over The Cited References

Applicant added claim 128 to make clear that the browser code generated by components can vary for different requests of the same document template. Claim 128 is dependent from claim 114 and is therefore patentable over the cited references for all the same reasons. In

addition, D'Arlach generates browser code during publishing and consequently delivers all the same browser code for every request as long as the document template is not changed.

Claims 115, 119, 121 and 123 were amended to change their dependency from claim 114 to claim 128.

IV. INFORMALITIES

The Examiner objected to claim 73 for improper syntax. Applicant has corrected this problem by amending claim 73 as shown in the Appendix.

The Examiner also rejected claim 83 under Section 112, paragraph 2, for insufficient antecedent basis for the term "final application." Applicant has corrected this problem by amending claim 83 as shown in the Appendix.

V. CONCLUSION

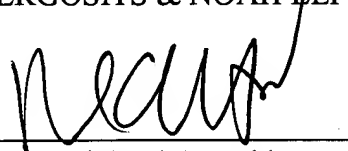
For all the foregoing reasons, applicant submits that the claims are in condition for allowance, and the Examiner's reconsideration to that end is respectfully solicited. The Examiner is encouraged to telephone the undersigned should additional issues remain.

Respectfully submitted,

DERGOSITS & NOAH LLP

Dated: January 13, 2005

By: _____



Richard A. Nebb
Reg. No. 33,540

Four Embarcadero Center, Suite 1450
San Francisco, California 94111
(415) 705-6377 tel
(415) 705-6383 fax
rnebb@dergnoah.com

APPENDIX

APPENDIX

Please amend claims 1, 42, 51-52, 59, 73-74, 78, 80-81, 83-85, 87-88, 92-94, 115, 119, 121, and 123, as shown below, and add new claim 128.

1. (currently amended) A software development system for applications that run on a data network which couples a server computer and a client computer, wherein the client computer runs a browser program, and whereupon request by the browser, an application generates generated documents for display by the browser and responds to the request with the generated documents pages comprising

a page generator running one of the an applications being developed and by generating the sending generated documents ~~to the browser for display as pages~~ including additional editing features for interpretation by the browser program;

an editor directly operating on the pages displayed by the browser via the editing features, thereby allowing the user to work on a functional application during development.

2. (previously amended) A software development system as claimed in claim 1, further comprising a plurality of components, and wherein developed applications comprise at least one page template capable of containing components, and wherein the editor provides features to insert, modify and delete a component on at least one page template, and wherein the page generator executes selected components on page templates.

3. (previously amended) A software development system as claimed in claim 2, wherein at least one of the components reacts interactively on user input by executing instructions of said component on the server.

4. (original) A software development system as in claim 3, wherein at least one of the components contains at least one other component.

APPENDIX

5. (previously amended) A software development system as in claim 3, wherein the set of components on pages generated from at least one page template can vary for different requests of said page template.

6. (previously amended) A software development system for use in a data network which couples a server computer to a client computer, wherein the client computer includes a first software program for generating a request for one or more pages from the server computer and for displaying pages, and wherein the server computer includes a second software program for receiving and processing the request from the client computer, for generating and storing pages, and for transmitting pages to the client computer in response to requests, the server computer further comprising:

- a data store,

- a plurality of components residing in the data store, including at least one component that reacts interactively on user input by executing instructions contained in said component on the server;

- a plurality of page templates residing in the data store, at least one page template having at least one selected component incorporated therein; and

- a server processor controlled by a third software program, said program providing instructions for selecting a page template based on the request from the client computer and instructions for generating a page from the page template for transmission to the client computer.

7. (original) The development system of claim 6, further comprising a component editor controlled by a fourth software program, said program providing instructions for interactively editing selected components on a page template.

8. (original) The development system of claim 6, wherein a component is nested within a component.

9. (previously amended) A method for generating documents for display by a browser using components that react interactively on user input by executing instructions

APPENDIX

on a server, comprising the following steps for execution on the server upon a document request:

assigning a unique identifier to at least one of the components; and
embedding the unique identifier into a generated document.

10. (previously amended) The method of claim 9, further comprising storing data on the server representing at least one of the components.

11. (previously amended) The method of claim 10, further comprising:
analyzing the request sent by the browser for unique identifiers; and
calling a function for the interactive components referenced by at least one of the unique identifiers contained in the request.

12. (previously amended) The method of claim 11, wherein at least one of the components is contained on a document template.

13. (original) The method of claim 11, wherein at least one of the components is called by a program.

14. (original) The method of claim 11, wherein at least one of the components is called by another component.

15. (original) The method of claim 11, wherein the data is stored into an object of an object oriented programming language and wherein the function is a method of the object.

16. (original) A method for implementing client server applications, comprising:
storing data objects on a server and assigning a unique identifier to each data object;
dynamically generating a document with the unique identifier embedded in the document; and

APPENDIX

analyzing requests for unique identifiers and calling at least one function for a data object associated with one of the unique identifiers found in the request.

17. (original) The method of claim 16, wherein the unique identifier is embedded inside a uniform resource locator contained in a tag of the document.

18. (original) The method of claim 16, wherein the unique identifier is embedded in scripts contained in the document.

19. (original) The method of claim 16, wherein the unique identifier is unique within a single session.

20. (previously amended) The method of claim 16, wherein the unique identifier is unique within all documents generated by a single server within a defined time period.

21. (original) The method of claim 16, wherein the data objects are created by an object-oriented programming language and said function is a method of one of these objects.

22. (previously amended) A computer running an application to develop and maintain applications using a web browser, comprising:

an editor operable within the web browser for inserting, deleting, and modifying components on document templates; and

a document generator for processing document templates, executing components and for generating documents from the document templates that are understandable by the web browser.

23. (previously amended) A computer as in claim 22, wherein the editor operates functional applications in an edit mode permitting editing directly in the web browser.

APPENDIX

24. (previously amended) A computer as in claim 23 wherein at least one of the components contains instructions and can react on subsequent document requests containing user responses by executing selected instructions of said component.

25. (previously amended) A computer as in claim 24, wherein the computer further comprises:

a store of component classes, each component class implementing one component kind; and

a parser able to detect components marked on document templates;

wherein the document generator works upon a document request using component classes to generate browser code; and

wherein the editor is capable of showing a menu of components for insertion into the document templates.

26. (previously amended) A system to modify documents on a server in a data network which couples said server computer to a client computer, the server computer comprising:

a document store;

a first software program including instructions for transforming at least one first document retrieved from the document store into a second document having features which permit editing of the first document such that at least a part of the second document appears and functions similar to the first document; and

a second software program including instructions to receive information from the client computer and instructions to modify the first document stored in the document store.

27. (previously amended) The system of claim 26, wherein the first document includes at least one component being executed by the first software program .

APPENDIX

28. (previously amended) The system of claim 27, wherein the second document includes handles and choosing one of the handles selects a component for an editing operation.

29. (previously amended) The system of claim 28, wherein at least one handle indicates the position of at least one component contained in the first document and said editing operation includes modifying the component, deleting the component, and displaying information regarding the component.

30. (previously amended) The system of claim 26, wherein the features include scripts.

31. (previously amended) The system of claim 30, wherein the scripts encapsulate information from the first document.

32. (original) The system as in claim 26, wherein the features incorporate information regarding the first document into the second document.

33. (previously amended) A system as in claim 32, wherein the information incorporated into the second document is used on the client computer in order to send change requests for the first document to the server.

34. (previously amended) A method for generating a document for display in a browser from a document template containing components, comprising:

for each component denoted on the document template, identifying a component class of the component; and

based on data contained in a request initiated by the browser storing a first object of the component class, the first object representing the component.

APPENDIX

35. (previously amended) The method of claim 44, further comprising calling a method of said component class to generate browser code, said method being the constructor.

36. (original) The method of claim 34, further comprising, for all components having a name attribute, looking up the component object in session memory based on said name attribute.

37. (previously amended) The method of claim 34, further comprising, for at least one component kind, for all components denoted on the document template having said component kind;

- generating a unique identifier;
- assigning said unique identifier to said object, and
- embedding said unique identifier into the browser code.

38. (previously amended) The method of claim 37, further comprising:
inserting objects for the components of at least said component kind into a list of listening components;

- working through all objects stored in the list of listening components whose unique identifier occurs inside a name in the form data set; and
- calling a method of at least one of these objects.

39. (previously amended) The method of claim 34, wherein the document template is parsed into a list of nodes, including text and component nodes, said method further comprising:

- determining if the current node is text or a component;
- if component, then calling a method for the component, comprising:
 - evaluating the attributes of the component if necessary;
 - identifying the component class associated with the component; and
 - calling the constructor method of the component class,
- said constructor method generating browser code;
- if text, then generating the text; and
- repeating these steps for each node.

APPENDIX

40. (original) The method of claim 39, wherein at least one component contains nested components and the method of claim 39 is recursively performed for all nodes nested inside the component.

41. (previously added) A software development system as in claim 1, the editor comprising a client part for execution on the client computer.

42. (currently amended) A software development system as in claim 41, wherein the client part comprises instructions for execution during editing that are automatically downloaded from the server in a request prior to editing.

43. (previously added) A system as in claim 26, additionally comprising at least one script for automatic download to the client that works in cooperation with the second document to permit editing of the first document.

44. (previously amended) The method of claim 34 wherein storing the first object comprises creating a new object as necessary.

45. (previously amended) The method as in claim 34 wherein components are denoted on document templates using tag syntax.

46. (previously added) The method as in claim 45 wherein the tag name identifies a component class.

47. (previously amended) The method as in claim 36 wherein components are denoted on document templates using tag syntax, wherein the tag name identifies a component class.

48. (previously added) The method as in claim 36 wherein the component object, if found, is reused to store the first object.

APPENDIX

49. (previously added) The method as in claim 36 wherein in case a component has a name attribute but no component object is found a new object is created and stored under said name in session memory.

50. (previously added) The method as in claim 49 wherein new objects are created for all components not having a name attribute.

51. (currently amended) A system for editing components on web document templates for use with a first software program including first instructions for generating a document request to obtain at least one generated document from a second software program and for displaying the generated document, the second software program capable of receiving and processing the document request and of transmitting first documents to the first software program in response to requests, said system comprising:

a plurality of components ~~encapsulating~~ containing instructions to generate browser code,

a plurality of document templates,

the second software program transmitting, while processing selected requests, second documents to the first software program that make the first software program display a user interface for editing functions used for maintaining components on document templates,

a third software program used by the second software program while processing selected document requests, the third software program including third instructions for modifying document templates in order to perform said editing functions.

52. (currently amended) The system of claim 51, wherein at least some components include fourth program instructions including steps to generate browser code for transmission to the first software program in response to a request from the first software program, wherein the browser code can differ for multiple requests for the same document template.

APPENDIX

53. (previously added) A system in claim 52 running on a data network, coupling a server computer and a client computer, the first program running on the client computer, the second program running on the server computer.

54. (previously amended) A system in claim 52 wherein second documents include HTML pages with embedded scripts.

55. (previously amended) The system of claim 52, wherein the edit function includes adding a component to a document template, removing a component from a document template, and modifying a component on a document template.

56. (previously added) The system of claim 52, further comprising a fifth software program used by the second software program while processing selected document requests, the fifth software program including fifth instructions for generating generated documents from document templates thereby calling fourth program instructions.

57. (previously added) The system of claim 56, wherein the generated document includes, if requested in edit mode, edit features for interpretation by the first software program.

58. (previously amended) The system of claim 56 further comprising instructions to allow the user to click on the generated document to select items to perform edit functions on.

59. (currently amended) A software development system for developing dynamic web documents, said dynamic web documents operating by being transformed into an end user's view upon a request by a web browser, the end user's view being provided to the browser in response to the request, comprising:
an editor program for editing dynamic web documents,

APPENDIX

a document generator for generating generated documents from dynamic web documents which look and function similar to the end user's view of the documents,

the editor program comprising first instructions for requesting the document generator to process a dynamic web document leading to a generated document,

the system further comprising second instructions for displaying at least some information items contained on said generated document in a view which allows the user to select an item to which a modification function will be applied,

the editor program further comprising third instructions to modify the dynamic web document to perform said modification function.

60. (previously added) The software development system of Claim 59 running on a data network which couples a server computer and a client computer, the document generator running on the server computer, the editor at least partly running on the client computer.

61. (previously amended) The software development system of claim 60 further comprising fourth instructions for execution during document generation to collect edit-information for use by the editor.

62. (previously added) The software development system of claim 60, wherein the editor uses a web browser for displaying said view.

63. (previously added) The software development system of claim 60, able to automatically repeat requesting the document generator to process the dynamic web document if required.

64. (previously amended) The software development system of Claim 59 further comprising a plurality of components including at least one component marked on said dynamic web document and including instructions for use by the document generator to generate browser code.

APPENDIX

65. (previously added) The software development system of claim 64, wherein the editor uses a web browser for displaying said view.

66. (previously amended) The software development system of claim 64, wherein modification functions include insert of a component, delete of a component, and modify a component.

67. (previously added) The software development system of claim 59, wherein said view looks, except for editing features, similar to the end-user view of the generated document.

68. (previously amended) The software development system of claim 59 further comprising sixth instructions to collect edit-information for use by the editor, said sixth instructions for execution during document generation.

69. (previously added) The software development system of claim 68, wherein the editor uses the edit-information to correctly modify the dynamic web document.

70. (previously amended) The software development system of claim 69, further comprising a plurality of components wherein the edit-information comprises position information on selected components marked on the .dynamic web document.

71. (previously added) The software development system of claim 59, wherein the editor uses a web browser for displaying said view.

72. (previously added) The software development system of claim 71, wherein first instructions comprise seventh instructions for initiating a reload in the browser.

73. (currently amended) The software development system of claim 59 wherein the editor program further comprises ~~ing~~ eighth instructions to display information on at least one element of at least one dynamic web document, that is replaced during

APPENDIX

document generation, without requesting the document generator to regenerate a the generated document.

74. (currently amended) A software development system for document templates that are intended for transformation into generated documents for display by a first software program, the first software program including first instructions for generating a document request to obtain at least one generated document and for displaying the generated document, comprising:

a plurality of components, that include instructions to generate browser code for transmission to the first software program,

an editor capable of performing edit functions maintaining components on document templates, the components capable to cooperate with the editor,

a plurality of document templates having components denoted thereon, and

a document generator comprising second instructions to, upon a document requests, generate generated documents from at least one document template for display by the first software program wherein the set of components on the generated documents can vary for different document requests for said document template.

75. (previously added) The software development system as in claim 74, wherein edit function comprises adding a component, modifying a component, and deleting a component.

76. (previously amended) The software development system as in claim 74, wherein tag syntax is used to denote at least one component on at least one document template, whereby the tag name identifies the component kind.

77. (previously added) The software development system of claim 74 running on a data network, which couples a server computer and a client computer, the document generator running on the server computer the editor running, at least partly, on the client computer.

APPENDIX

78. (currently amended) The software development system as in claim 74, wherein at least one component, that can react interactively on subsequent document requests, can upon selected document requests for said document template be excluded from the said generated document.

79. (previously added) The software development system as in claim 78 further comprising third instructions to prevent excluded components from reacting on subsequent document requests.

80. (currently amended) A software development system as in claim 79, said third instructions comprising fourth instructions to, upon a first document request, store information in session memory on some of the components, that are present on the document generated based on the first request document, and fifth instructions to, upon subsequent document requests, only react on components that have been remembered in session memory thereby avoiding tampering with excluded components on the side of the first program.

81. (currently amended) A software development system as in claim 74 wherein at least one first component contains sixth instructions to decide upon a request for said document template about exclusion of components nested inside the first component from the generated document.

82. (previously amended) A software development system as in claim 74 the system able to provide an editable view taking the varying set of components into account.

83. (currently amended) A software development system as in claim 74 the system able to provide an editable view that includes and excludes selected components

APPENDIX

on different requests for said document template similarly as the ~~final application~~ end user's view of the document template.

84. (currently amended) A software development system as in claim 74 wherein ~~a the-generated~~ document generated for of at least one document template contains more components than the document template for at least one document request.

85. (currently amended) The software development system as in claim 74, wherein multiple instances of at least one third component denoted on the document template can be included in at least one of the ~~generated~~ documents generated from said document template.

86. (previously amended) The software development system as in claim 74, further comprising seventh instructions to assign a unique identifier to each component instance of at least one seventh component, whereby the seventh component includes eighth instructions to qualify names generated into the browser code with the unique identifier.

87. (currently amended) A software development system as in claim 74, wherein at least one fourth component contains ninth instructions to decide upon a request about how many instances of components nested inside the fourth component are included ~~into the-generated document~~ in the documents generated from said document template.

88. (currently amended) A software development system as in claim 74 the editor able to provide an editable view that includes multiple instances of selected components similarly as the ~~final application~~ end user's view of the document template.

89. (previously amended) A software development system as in claim 74 wherein at least one sixth component includes tenth instructions to display the sixth component, the tenth instructions being used to generate browser code for displaying the sixth component during editing as well as during normal use of the component.

APPENDIX

90. (previously amended) An editor for use with a web browser, the editor allowing the user to edit at least one document displayed by the browser, wherein scripts contained in said document remain functional during editing, the editor including a first software program for execution within the browser and for processing selected clicks on the view of said document displayed in the browser by initiating editing functions .

91. (previously added) The editor as in claim 90 using at least two windows, a first browser window displaying said document and a second window for displaying information on an element contained in said document.

92. (currently amended) The editor in claim 90 further comprising a second software program for storing modifying indications on said document in cooperation with the first software program.

93. (currently amended) The editor as in claim 92 further comprising a third program for transforming an original ~~the~~ document into ~~a-generated~~ the document, the browser displaying the ~~generated~~ document as said view looking similar to the original document and interpreting editing features contained in the ~~generated~~ document.

94. (currently amended) The editor as in claim 93 wherein said original document is a dynamic document having components denoted thereon, the third software program further comprising instructions for generating browser code in cooperation with selected instructions contained in the components.

95. (previously added) The editor as in claim 94 wherein the browser together with the first software program is running on a client computer connected to a server computer via a data network, wherein the second and the third software program run on the server computer.

APPENDIX

96. (previously added) The editor in claim 90 wherein links contained in said document stay functional allowing the user to browse and edit at the same time.

97. (previously amended) A system for displaying dynamically generated documents in a data network coupling a server computer to a client computer, wherein the client computer has a first software program including first instructions for generating a document request to obtain at least one generated document from the server computer and for displaying the generated document, comprising:

a plurality of components for execution on the server computer, including a first component including second program instructions to generate browser code and third program instructions for execution on the server which are initiated by the user interacting with the first component, and,

fourth program instructions on the server computer for, based on data contained in a document request initiated by the first software program on the client computer, generating generated documents for transfer to the client computer and display by the first software program, thereby calling second program instructions of components.

98. (previously added) The system of claim 97, the server computer further comprising fifth program instructions for analyzing said data and for calling third program instructions of first component as necessary.

99. (previously added) The system of claim 98, further comprising a plurality of document templates residing in the data store, at least one of the document templates having at least one second component denoted thereon.

100. (previously added) The system of claim 99, wherein tag syntax is used to denote the second component, whereby the tag name identifies a component.

101. (previously added) The system of claim 99, wherein at least one third component denoted on a document template contains the first component.

APPENDIX

102. (previously added) The system of claim 101, wherein third components implementation scheme includes logic to decide how often to insert the first component into the generated document.

103. (previously added) The system of claim 98, the server computer further comprising sixth program instructions for, based on the document request, deciding to insert more than one instance of the first component into the generated document.

104. (previously added) The system of claim 103, making sure that multiple instances of the first component do not interfere by qualifying names generated into the browser code using unique identifiers.

105. (previously added) The system of claim 98, the server computer further comprising seventh program instructions for, based on the data, deciding to exclude the first component from the generated document.

106. (previously amended) The system of claim 105, wherein fifth instructions call third instructions only if the first component was contained on a page previously transferred to the client.

107. (previously added) The system of claim 98, wherein fifth program instructions include eighth program instructions to analyze said data for user interactions with multiple components and to call third program instructions of multiple components as necessary.

108. (previously added) The system of claim 107, wherein components include ninth instructions to check for errors and fifth instructions include tenth instructions to call ninth instructions of components as necessary and to suppress subsequent calling of third instructions in case of errors.

APPENDIX

109. (previously added) The system of claim 98, further comprising eleventh program instructions for storing a data object in session memory representing at least one component instance included in a generated document, said eleventh program instructions for execution while dynamically generating a document.

110. (previously added) The system of claim 109, wherein third program instructions are encapsulated in a method of data objects, fifth program instructions including twelfth program instructions for identifying the data object that represents a component instance the user interacted with and for calling said method of said data object as necessary.

111. (previously added) The system of claim 110, further including thirteenth instructions for deciding based on said data to include more than one instance of a component into the generated document.

112. (previously added) The system of claim 109, further comprising twelfth program instructions for assigning a unique identifier to the first component instance, for associating the unique identifier with the data object, and for including the unique identifier into the generated document, said twelfth program instructions for execution while dynamically generating a document .

113. (previously added) The system of claim 112, wherein fifth program instructions analyze said data for unique identifiers and include thirteenth instructions for identifying the associated data object.

114. (previously amended) A system for displaying dynamically generated documents in a data network coupling a server computer to a client computer, wherein the client computer has a first software program including first program instructions for generating a request to obtain at least one generated document from the server computer and for displaying the generated document, comprising:

APPENDIX

a plurality of components for execution on the server, at least one of the components including first features to cooperate with an editor in editing said component and second program instructions to generate browser code, and

third program instructions on the server for, based on the data contained in a request initiated by the client computer, generating generated documents for transfer to the client computer, thereby calling second program instructions of selected components.

115. (currently amended) The system of claim 128 ~~444~~ wherein first features include fourth program instructions for passing information to the editor.

116. (previously amended) The system of claim 115 wherein at least part of said information is collected during execution of selected components on the server.

117. (previously added) The system of claim 115 wherein said information is transmitted from the server to the client.

118. (previously amended) The system of claim 115 wherein said information includes attribute values of said component.

119. (currently amended) The system of claim 128 ~~444~~ wherein first features include fifth instructions that display additional editing features of the component during editing.

120. (previously added) The system of claim 119 wherein said editing features include handles.

121. (currently amended) The system of claim 128 ~~444~~ wherein first features include an extension for use by the editor, said extension for enabling editing of an attribute value of the components .

APPENDIX

122. (previously amended) The system of claim 121 wherein said extension enables display of a page for editing the components attribute values.

123. (currently amended) The system of claim ~~128~~ 114 wherein at least one component is denoted on at least one document template using tag syntax, whereby the tag name identifies a component kind.

124. (previously amended) The system of claim 114 containing at least one component wherein second program instructions are used to generate browser code for displaying the component during editing and during normal use.

125. (previously amended) A method for editing an application that is built using components and that operates by generating documents comprising the steps of:

running the application, thereby executing selected components and generating a generated document,

displaying a view of the generated document,

selecting a component by clicking on selected portions of said view,

identifying the selected component in the source code of the application,

initiating a modification function modifying the source code of the application.

126. (previously added) The method of claim 125 wherein the running step and the displaying step are repeated after applying a modification function.

127. (previously added) The method of claim 125 further comprising collecting edit information for use by the identifying step.

128. (currently added) The system of claim 114 additionally comprising a plurality of document templates with components denoted thereon, whereby the browser code generated by the components can vary for different requests of the same document template.